Aarhus Universitet
Datalogisk Institut
Ny Munkegade
8000 Århus C

18. december 2001

Arkitektur
Obligatorisk opgave

Obligatorisk opgave besvaret af Per Abrahamsen, årskort 2001358, hold 3, Mads Peter Lindberg, årskort 20001990, hold 3, og Søren S. Munk, årskort 20001614, hold 3.
www.daimi.au.dk/ smunk/dArkOS/obl
Besvarelsen omfatter 51 nummererede sider.

Enten: Rettidigt afleveret i den røde kasse udenfor Informationskontoret, R2, senest den 19. december 2001 kl. 12.00.
Eller: **Ikke** rettidigt afleveret den   /   til ______________.

Indhold

1 Indledning

Denne rapport er besvarelse af obligatorisk opgave i kurset Arkitektur. Vi har i ijvm implementeret imul, idiv, ishr, iushr, ishl og if_icmplt til kørsel på mic1. Disse er blevet afprøvet og ser ud il at virke tilfredstillede. Læseren forudsættes at kende opgaveformuleringen samt bogen Structured Computer Organization af Andrew Tanenbaum.

2 Ændringer i IJVM

De seks nye ordrer `imul`, `idiv`, `if_icmplt`, `ishl`, `ishr` og `iushr` er tilføjet IJVM og deres virkning er som beskrevet i nedenstående tabel.

Opcode	**Mnemonic**	**Description**
0x10	BIPUSH byte_exp	Push a byte onto stack
0x59	DUP	Copy top word on stack and push onto stack
0xA7	GOTO label	Unconditional jump
0x60	IADD	Pop two words from stack; push their sum
0x7E	IAND	Pop two words from stack; push Boolean AND
0x99	IFEQ label	Pop word from stack and branch if it is zero
0x9B	IFLT label	Pop word from stack and branch if it is less than zero
0x9F	IF_ICMPEQ label	Pop two words from stack and branch if they are equal
0x84	IINC varnum_exp, byte_exp	Add a constant value to a local variable
0x15	ILOAD varnum_exp	Push local variable onto stack
0xB6	INVOKEVIRTUAL method	Invoke a method
0x80	IOR	Pop two words from stack; push Boolean OR
0xAC	IRETURN	Return from method with integer value
0x36	ISTORE varnum_exp	Pop word from stack and store in local variable
0x64	ISUB	Pop two words from stack; push their difference
0x13	LDC_W constant_exp	Push constant from constant pool onto stack
0x00	NOP	Do nothing
0x57	POP	Delete word from top of stack
0x5F	SWAP	Swap the two top words on the stack
0xC4	WIDE	Prefix instruction; next instruction has a 16-bit index
0x68	imul	Pop two words from stack; Push their multiplication
0x6C	idiv	Pop two words of the stack; Divides the second with the top; Push the result
0xA1	if_icmplt label	compares the top two integers of the stack. If value1 is less than value2, it goes to the label.
0x78	ishl	Pops two ints off the stack. Shifts value1 left by the amount indicated in the five low bits of value2. The result is pushed on the stack.
0x7A	ishr	Pops two ints off the stack. Shifts value1 right by the amount indicated in the five low bits of value2. The int result is then pushed back onto the stack. value1 is shifted arithmetically
0x7C	iushr	Pops two ints off the operand stack. Shifts value1 right by the amount indicated in the five low bits of value2. The int result is then pushed back onto the stack.

3 Ændringer i `ijvm.mal`

De seks nye ordrer er blevet implementeret ved at ændre og tilføje i mikroprogrammet `ijvm.mal`. Ændringerne og tilføjelserne er vist i appendiks B. Implementationen af de seks nye ordrer er foretaget, så deres udførelse nøje svarer til udførelsen af de tilsvarende ordrer på JVM. I ugeopgave 41 blev de seks nye ordrer tilføjet den eksisterende IJVM simulator. Disse tilføjelserne er vist i appendiks A.

I de følgende afsnit beskrives i yderligere detaljer, hvordan hver af de seks nye ordrer er blevet tilføjet i mikroprogrammet.

3.1 Skifteoperationer – `ishl`, `ishr` og `iushr`

Venstreskift (`ishl`) er implementeret således:

```
ishl = 0x78:
        MAR = SP = SP - 1; rd
        OPC = 1 << 8
        OPC = OPC >> 1
        OPC = OPC >> 1
        OPC = OPC >> 1
        H = OPC = OPC - 1
        TOS = TOS AND H
ishl_test:
        Z = TOS; if (Z) goto ishl_end; else goto ishl_st
ishl_st:
        H = MDR
        MDR = MDR + H
        TOS = TOS - 1; goto ishl_test
ishl_end:
        TOS = MDR; wr
        goto main
```

```
Vi har brugt følgende til implementationen:
x ishl y:

y = (y AND 31)
while (y > 0){
x = x * 2
y = y -1}
```

Vi bruger OPC til at danne tallet 31 som vi AND'er med TOS som svarrer til vores y i højniveaualgoritmen. Så står vi med de fem mindstbetydende bit af den oprindelige TOS i TOS. Så går vi ind i vores while løkke, så længe TOS ikke er 0 så ganger vi MDR, som svarrer til vores x, med 2 ved at lægge MDR over i H og så lægge H = MDR ind i MDR, hvilket svarrer til at flytte bittene i MDR 1 til venstre. Vi tæller så TOS 1 ned. Når TOS så bliver 0 betyder det at vi har flyttet bittene i MDR y antal gange og vi lægger MDR på stakken.

Aritmetisk højreskift (`ishr`) er implementeret således:

```
ishr = 0x7A:
        MAR = SP = SP - 1; rd
        OPC = 1 << 8
        OPC = OPC >> 1
        OPC = OPC >> 1
        OPC = OPC >> 1
        H = OPC - 1
        TOS = H AND TOS

ishr_while:
        TOS = TOS - 1; if (N) goto ishr_endwhile; else goto ishr_do
ishr_do:
        MDR = MDR >> 1; goto ishr_while
ishr_endwhile:
        TOS = MDR; wr; goto main

Vi har brugt denne algoritme til implementationen:
x ishr y:
y = y AND 31
while (y > 0){
x >> 1
y = y - 1}
```

Vi bruger OPC til at lave tallet 31 som vi AND'er med TOS, som svarrer til vores y. Vi har så de fem mindste bit af y i TOS. Så Går vi ind i whileløkken hvor vi tester på om TOS er 0 hvis den ikke er nul laver vi 1 højreskift på MDR, som svarrer til vores x. Vi tæller samtidig TOS 1 ned. Når TOS så bliver 0 springer vi ud af løkken og skriver MDR til stakken.

Logisk højreskift (`iushr`) er implementeret således:

```
iushr = 0x7c:
        MAR = SP = SP - 1; rd
        OPC = 1 << 8
        OPC = OPC >> 1
        OPC = OPC >> 1
        OPC = OPC >> 1
        H = OPC - 1
        TOS = H AND TOS
iushr_test:
        Z = TOS; if (Z) goto iushr_end; else goto iushr_st
iushr_st:
        MDR = MDR >> 1
        TOS = TOS - 1
        N = MDR; if (N) goto iushr_st2; else goto iushr_test
iushr_st2:
        H = 1 << 8
        H = H << 8
        H = H << 8
        H = H >> 1
        H = H << 8
        MDR = MDR - H; goto iushr_test
iushr_end:
        TOS = MDR; wr; goto main
```

Vi bruger OPC til at skabe tallet 31 som vi AND'er med TOS, så har vi de fem mindste bit i TOS som nu er det antal gange vi skal skifte til højre.Vi går nu ind i vores whileløkke hvor vi tester på om TOS er 0. Hvis det ikke er tilfældet Skifter vi MDR 1 til højre og tæller TOS 1 ned. Vi tjekker så på om MDR er negativ altså om der er 1 i højeste bit. Hvis der er det så danner vi i H et bitmønster hvor der er 1 i højeste bit og 0 i resten, og den trækker vi så fra MDR, på den måde har vi sikret os at der nu bliver sat 0 ind i højeste bit, når vi skifter MDR. Vi springer nu tilbage og tjekker på TOS når den så er blevet 0 skriver vi MDR på stakken.

3.2 Aritmetiske operationer – `imul` og `idiv`

Multiplikation (`imul`) er implementeret således:

```
imul = 0x68:
        MAR = SP = SP - 1; rd
        H = TOS
        OPC = MDR
        MDR = 0
imul_test:
        Z = TOS; if (Z) goto imul_end; else goto imul_while
imul_while:
        H = 1
        Z = H AND TOS; if (Z) goto imul_lige; else goto imul_ulige
imul_lige:
        N = TOS = TOS >> 1; if(N) goto imul_fix; else goto imul_cont
imul_fix:
        H = 1 << 8
        H = H << 8
        H = H >> 1
        H = H << 8
        H = H << 8
        TOS = TOS + H
imul_cont:
        H = OPC
        OPC = OPC + H; goto imul_test
imul_ulige:
        H = OPC
        MDR = MDR + H; goto imul_lige
imul_invert:
        LV = MDR
        H = -H; goto imul_finish_end
imul_end:
        H = MDR; rd
        Z = LV; if (Z) goto imul_invert; else goto imul_finish_end
imul_finish_end:
        TOS = MDR = H; wr; goto main
```

```
Vi har brugt følgende algoritme:
x imul y:
z:= 0; u:= x; v:= y;
while u <> 0 do
begin
if odd(u) then z:= z + v;
u:= u div 2; v:= 2*v;
end
```

Vi kører en while så længe TOS, som svarrer til u, ikke er 0, og vi spørger så om TOS er ulige ved at spørge om TOS AND 1 er 0, hvis TOS AND 1 er 0 så er tallet lige og vi dividerer TOS med 2 og ganger OPC med 2. OPC svarer til vores v. Hvis TOS derimod var ulige lægger vi OPC til MDR, som svarer til vores z. Når TOS så på et tispunkt er blevet 0 springer vi ud af løkken og skriver MDR til stakken.

Division (`idiv`) er implementeret således:

```
idiv =  0x6c:
        MAR = SP = SP - 1; rd
        H = LV
        OPC = MDR
        MDR = H; wr
        MAR = SP = SP + 1
        H = LV = TOS
        MDR = CPP; wr
        empty
        MDR = CPP = 0
## test for nævner er nul
idiv_test_nul:
        Z = TOS; if (Z) goto idiv_nul; else goto idiv_test_equal
idiv_test_equal:
        Z = OPC - H; if (Z) goto idiv_facit1; else goto idiv_test_neg
idiv_test_neg:
        N = TOS; if (N) goto idiv_neg; else goto idiv_test1

idiv_neg:
        N = TOS = LV = H = -H; if (N) goto idiv_nul_branch; else goto idiv_neg_cont
idiv_nul_branch:
        goto idiv_nul
idiv_neg_cont:
        CPP = 1
        MDR = 0; goto idiv_test1
```

```
idiv_test1:
        N = H = H + TOS; if (N) goto idiv_while1_end; else goto idiv_while1
idiv_while1:
        TOS = H; goto idiv_test1
idiv_while1_end:
        H = TOS; goto idiv_test2
idiv_test2:
        N = OPC; if (N) goto idiv_neg2; else goto idiv_while3
idiv_neg2:
        H = OPC
        N = OPC = -H; if (N) goto idiv_neg3; else goto idiv_neg4
idiv_neg4:
        CPP = CPP - 1; goto idiv_while3
idiv_neg3:
        CPP = CPP - 1
        H = TOS
        MDR = MDR + 1
        OPC = OPC - H; goto idiv_while3
idiv_test4:
        H = TOS
        Z = LV - H; if (Z) goto idiv_end2; else goto idiv_st3
idiv_st3:
        TOS = TOS >> 1
        H = MDR
        MDR = MDR + H
idiv_while3:
        H = TOS
        N = OPC - H; if (N) goto idiv_st2; else goto idiv_st
idiv_st2:
        H = TOS; goto idiv_test4
idiv_st:
        MDR = MDR + 1
        OPC = OPC - H; goto idiv_test4
idiv_facit1:
        MDR = 1; goto idiv_end2
idiv_nul:
        goto idiv_end2
idiv_end2:
        H = MDR; rd
        Z = CPP; if (Z) goto idiv_finish_end; else goto idiv_invert
idiv_invert:
        H = - H; goto idiv_finish_end
```

```
idiv_finish_end:
        MAR = SP = SP - 1
        CPP = MDR; rd
        empty
        LV = MDR
        TOS = MDR = H; wr; goto main
```

```
Vi har brugt følgende algoritme som udgangspunt:
begin r:=x; q:=0; w:=y;
             while (w <= r) do w := 2*w;
             { w = 2^n * y > x }
             while w <> y do
             begin { q*w + r = x, r>= 0 }
               q:= 2*q; w:= w div 2;
               if w <= r then
               begin r:= r - w; q:= q + 1
               end
             end
             { q*y + r = x, 0 <= r <  y; q = x div y }
       end
```

```
Dog har vi for at gøre koden perfekt brugt:
While (2*w >= 0) do w:= 2*w
While (1) do{
    if (w <= r ) do{
       r:= r - w;
       q = q + 1;
       };
    if ( w == y ) do break;
    w:= w/2;
    q:= 2*q
};
```

Vi gemmer LV og CPP på stakken, så vi kan bruge disse registre igennem algoritmen. I LV gemmer vi vores y som er det samme som TOS i begyndelsen. CPP bruger vi til at holde styr på om der skal skiftes fortegn til sidst, altså om x og y har forskelligt fortegn. TOS svarer til vores w, OPC til vores r og MDR til vores q. Først tester vi om TOS er lig 0 for hvis den er det så er vi færdige og vi har så valgt at hvis man prøver at dividerer med 0 så ligger vi 0 på stakken. Her kunne vi have valgt at kalde mic1_exit, der stopper programkørslen. Vi prioriterede at programmet skulle fortsætte med at køre, selvom der opstår fejl
Dernæst tester vi om TOS og OPC er ens, hvis de er det sætter vi MDR lig 1 og

skriver det til stakken, efter at have hentet CPP og LV's oprindelige værdier og lagt dem i CPP og LV.
Er de ikke ens tester vi om TOS er negativ, er den det negerer vi både TOS og LV, der endnu er ens. Så tester vi igen om TOS er negativ for hvis den er det, er den lig det -2^{31} og det tilfælde behandler vi som at dividerer med 0 da resultatet kun er forskelligt fra nul hvis tælleren også er -2^{31}, og det har vi allerede testet for tidligere i algoritmen.
Hvis negationen har gjort TOS positiv så sætter vi CPP til 1. Vi går nu ind i en whileløkke, hvor vi så længe 2*TOS er positivt ganger TOS med 2, dvs. når TOS ikke længere kan fordobles uden der sker overløb forlades løkken. Der kan maksimalt komme 30 gennemløb i denne lykke
Vi tjekker så om OPC er negativ(idiv_test2), hvis det er tilfældet negeres den (idiv_neg2) og CPP ændres(idiv_neg4). Hvis OPC efter negeringen stadig er negativ er den -2^{31} og man betragter -2^{31} som unsigned. Så kan man trække nævneren fra og så er man sikker på at man har et positivt 2-kompliment tal. Herefter kan vi med god samvittighed ændre CPP, der senere bruges til at tjekke om der skal skiftes fortegn(idiv_invert).
Nu har vi taget hensyn til alle special-tilfælde og fortsætter til vores divisions-algoritme(idiv_while3), der er beskrevet ovenfor(s.9). Her starter algoritmen med at teste om TOS - OPC > 0(w <= r) og er det tilfældet trækkes TOS fra OPC (r = r - w) og MDR tælles 1 op(q = q + 1)(idiv_st). Ligemeget om TOS - OPC > 0(w <= r) er sand eller ej fortsætter vi med at teste om TOS = LV(w = y). Hvis de ikke er lig hinanden halveres TOS(w = w/2) og MDR fordobles(q = 2q). Når de bliver lig hinanden er vi færdige og vi mangler kun at negere resultatet i de tilfælde hvor kun den ene var negativ(CPP = 0)(idiv_invert) og ordne registrene, så både LV og CPP får deres værdier indlæst fra stakken. Resultat gemmes i TOS og skrives til stakken.
Vi er sikre på at vores algoritme virker for alle 2-komplement-tal, der kan repræsenteres med 32 bit. Det har vi gjort ved at tage hensyn til alle specialtilfælde, fra division med 0 til de mere specifikke f.eks. -2^{31}'s negation, der som bekendt stadig er negativ(sig selv) og derfor er et specialtilfælde.

3.3 Betinget hop – `if_icmplt`

Det betingede hop (`if_icmplt`) er implementeret således:

```
if_icmplt = 0xA1:
        MAR = SP = SP - 1; rd
        MAR = SP = SP - 1
        OPC = MDR; rd; if (N) goto xNeg; else goto xPos
xPos:
        H = TOS; if (N) goto if_icmpltF; else goto if_icmpltPOS
xNeg:
        H = TOS; if (N) goto if_icmpltNEG; else goto if_icmpltT
if_icmpltT:
        TOS = MDR; goto T
if_icmpltF:
        TOS = MDR; goto F
if_icmpltPOS:
        TOS = MDR; goto if_icmplt_end
if_icmpltNEG:
        TOS = MDR; goto if_icmplt_end
if_icmplt_end:
        N = OPC - H; if (N) goto T; else goto F

Vi har brugt denne algoritme til implementationen, hvor vi spørger er
x mindre end y:
if ( x >= 0 And Y >= 0 ) {
    if( (x - y) < 0) {
         return TRUE;}
    else return False;
elseif( x < 0 And Y < 0 ) {
    if( (x - y) < 0) {
         return TRUE;}
    else return False;
elseif ( x < 0 and y >= 0){
    return True}
elseif ( x >= 0 and y < 0){
    reurn FALSE}
```

Vi bruger OPC som x og TOS som y, og vi tjekker først om OPC er negativ, og derefter tjekker vi om TOS er negativ, hvis de har samme fortegn tester vi om OPC - TOS er negativ, hvis den er det så er OPC mindre end TOS og vi går til T, og hvis ikke går vi til F.
Har de ikke samme fortegn dvs. hvis OPC er ikke negativ og TOS er negativ går vi til F, men hvis OPC er negativ og TOS er ikke negativ går vi til T.

4 Afprøvning

Det udvidede mikroprogram `ijvm2.mal` er afprøvet med henblik på følgende:

- De gamle ordrer virker som før. Dette er afprøvet ved at køre programmet `test.j` fra IJVM manualen. Kildeteksten til dette program er vist i appendiks C. Kørslerne med programmet er ligeledes vist i appendiks C.
- En intern afprøvning af mikroprogrammet, der sikrer at alle de tilføjede mikroordrer er afprøvet mindst én gang og at alle de tilføjede betingede hop er afprøvet, så betingelsen har været både sand og falsk. Dette er afprøvet med de testprogrammer som er vist i appendiks D. Disse testprogrammer er konstrueret i overensstemmelse med nedenstående tabel, der angiver valg af testdata. Kørslerne med testprogrammerne er ligeledes vist i appendiks D.
- En ekstern afprøvning af mikroprogrammet, der sikrer, at de nye ordrer virker som på IJVM2 simulatoren. Dette er afprøvet med de testprogrammer som er vist i appendiks D. Kørslerne er ligeledes vist i appendiks D.

Vi har for hver ordre kigget på de betingede hop og udfra hvad der sker før vi kommer til betingelsen valgt nogle tal der sørger for at vi kommer til begge labels der er efter betingelsen.
Hvis man kigger på koden for vores ordre, vil man kunne se at hvis blot man har været gennem alle betingelserne og her har haft både sandt og falsk resultat, så har man også været gennem alle mikroordrene.
De testdata vi har brugt har udover de tal der er angivet i tabellen nedenfor også andre tal, så vores testprogrammer er faktisk blevet brugt både til den interne og den eksterne test.

Nedenstående tabel angiver for alle tilføjede betingede hop i mikroprogrammet, hvilke testdata, der er anvendt for at sikre, at det betingede hop har været udført så betingelsen har været både sand og falsk. Valg af testdata i appendiks D er foretaget systematisk udfra denne tabel.

Ordre	Betinget hop	Test tilfælde/data	Resultat
ishr	TOS = TOS - 1; if (N) goto ishr_endwhile; else goto ishr_do	N=0: a = 4, b = -1 N=1: a = 0, b = 1	-1 1
imul	Z = TOS; if (Z) goto imul_end; else goto imul_while	Z=0: a = 53, b = 3 Z=1: a = 0, b = 53	159 0
imul	Z = H AND TOS; if (Z) goto imul_lige; else goto imul_ulige	Z=0: a = 2, b = 53, H=1 Z=1: a = 3, b = 53, H=1	106 159
imul	N = TOS = TOS » 1; if(N) goto imul_fix; else goto imul_cont	Z=0: a = 2, b = 53 Z=1: a = -3, b = 53	106 -169
idiv	Z = TOS; if (Z) goto idiv_nul; else goto idiv_test_equal	Z=0: a = 1, b = 42 Z=1: a = 0, b = 42	42 0
idiv	Z = OPC - H; if (Z) goto idiv_facit1; else goto idiv_test_neg	Z=0: a = 1, b = 42 Z=1: a = -2^{31}, b = -2^{31}	42 1
idiv	N = TOS; if (N) goto idiv_neg; else goto idiv_test1	N=0: a = 2, b = 42 N=1: a = -2, b = 42	21 -21
idiv	N=TOS=LV=H=-H;if(N)goto idiv_nul_branch; else goto idiv_neg_cont	N=0: a = -2, b = 42 N=1: a = -2^{31}, b = 42	-21 0
idiv	N = H = H + TOS; if (N) goto idiv_while1_end; else goto idiv_while1	N=0: a = 2, b = 42 N=1: a = $2^{31} - 1$, b = 42	21 0
idiv	N = OPC; if (N) goto idiv_neg2; else goto idiv_while3	N=0: a = 4, b = 42 N=1: a = 7, b = -42	10 -6
idiv	N = OPC = -H; if (N) goto idiv_neg3; else goto idiv_neg4	N=0: a = 7, b = -42 N=1: a = 2^{31}-1, b = -2^{31}	-6 -1
idiv	Z = LV - H; if (Z) goto idiv_end2; else goto idiv_st3	se pkt. 1	
idiv	N = OPC - H; if (N) goto idiv_st2; else goto idiv_st	N=0: a = 1, b = 2^{31}-1 N=1: a = 2, b = 42	2^{31}-1 21
idiv	Z = CPP; if (Z) goto idiv_finish_end; else goto idiv_invert	Z=0: a = 2, b = -42 Z=1: a = 2, b = 42	-21 21
ishl	TOS = TOS - 1; if (N) goto ishl_end; else goto ishl_st	N=0: a = 2, b = 1 N=1: a = 0, b = 2	4 2
iushr	Z = TOS; if (Z) goto iushr_end; else goto iushr_st	Z=0: a = 20, b = -3 Z=1: a = 0, b = 1	4095 1
iushr	N = MDR; if (N) goto iushr_st2; else goto iushr_test	N=0: a = 4, b = 2 N=1: a = 20, b = -3	0 4095
if_icmplt	OPC = MDR; rd; if (N) goto xNeg; else goto xPos	N=0: a = 2, b = 4 N=1: a = 2, b = -4	False True
if_icmplt	H = TOS; if (N) goto if_icmpltF; else goto if_icmpltPOS	N=0: a = 4, b = 2 N=1: a = -2, b = 4	True False
if_icmplt	H = TOS; if (N) goto if_icmpltNEG; else goto if_icmpltT	N=0: a = 6, b = -8 N=1: a = -6, b = -3	True False
if_icmplt	N = OPC - H; if (N) goto T; else goto F	N=0: a = 3, b = 7 N=1: a = -3, b = -7	False True

Pkt 1: Denne if svarer til algoritmens 'while w <> y do' som vi altid vil komme ind i mindst en gang, og at den hopper ud af den igen er også indlysende da, w starter som y og vi ganger så w med 2 et vist antal gange og derefter dividerer vi w med 2 for hvert gennemløb af løkken.

5 Udførelsestider for IJVM2 ordrer.

For de 20 eksisterende ordrer fås følgende udførelsestider målt i maskincykler:

Ordre	**Antal maskincykler**
bipush	4
dup	3
goto	7
iadd	4
iand	4
ifeq	5
iflt	5
if_icmpeq	7
iinc	7
iload	6
invokevirtual	23
ior	4
ireturn	9
istore	7
isub	4
ldc_w	5
nop	2
pop	4
swap	7
wide	2

For de seks nye ordrer fås følgende udførelsestider målt i maskincykler:

Ordre	**Antal maskincykler**
imul	$7 + 3\frac{1}{3}x$
idiv	$50\frac{2}{13} + 9(\frac{x}{y})$
ishl	11 + 4n
ishr	10 + 2n
iushr	$14\frac{2}{3} + \frac{2}{3}$n
if_icmplt	$8\frac{1}{2}$

Af de nye er det kun if_icmplt der ikker afhænger af de brugte værdier.
imul afhænger af x som svarer til vores value1, vi har tre tilfælde i imul, og det angivede er et gennemsnitlige antal maskincykler, worst case er 7+14x.
I idiv har vi igen givet et gennemsnit, mens worstcase hedder $89 + 9\frac{x}{y}$. Hvor x er value2 og Y er value1.
I ishl er n value1 altså det antal gange vi vil skifte value2 til højre dette tal er fast det tager lige meget hvad 11 + 4n.
I ishr er n value2 igen det antal pladser der skal skiftes, det samme er tilfældet for iushr.

6 Afslutning

Vi mener at vi hermed har løst denne opgave til fulde, da vi nu ved brug af mic1 kan løse samme problemer som vi kunne med vores ijvm, og med samme resultat. Dog er der den forskel at vi har valgt ikke at kaste en exception hvis man prøver at dividere med 0, men istedet smider vi et 0 på stakken.
Vi har haft nogle småproblemer med mic-1 assembleren, som dog forsvandt ved re-assembling.
Dette er vi kommet frem til ved en konklussion om at vi naturligvis ikke har begået nogen fejl...

A IJVM2 simulator

C-programmet som implementerer IJVM simulatoren blev i ugeopgave 40 ændret ved at følgende er blevet tilføjet:

Nedenunder ses vores tilføjelser til ijvm.c:

```
case IJVM_OPCODE_IMUL:
         a = ijvm_pop (i);
         b = ijvm_pop (i);
         ijvm_push (i, a * b);
         break;

       case IJVM_OPCODE_IDIV:
         a = ijvm_pop (i);
         b = ijvm_pop (i);
         ijvm_push (i, b / a);
         break;

       case IJVM_OPCODE_IF_ICMPLT:
         offset = ijvm_fetch_int16 (i);
         a = ijvm_pop (i);
         b = ijvm_pop (i);
         if (a < b)
           i->pc = opc + offset;
         break;

       case IJVM_OPCODE_ISHL:
         a = ijvm_pop (i);
         b = ijvm_pop (i);
         a = a & 31;
         ijvm_push (i, b << a );
         break;

       case IJVM_OPCODE_IUSHR:
         a = ijvm_pop (i);
         b = ijvm_pop (i);
         a = a & 31;
         ijvm_push (i,((uint32) b) >> a );
         break;

       case IJVM_OPCODE_ISHR:
```

```
        a = ijvm_pop (i);
        b = ijvm_pop (i);

        a = a & 31;
        if ( b < 0 ) {
          b = ~b;
          b=b>>a;
          b = ~b;
        }
        else {
          b=b>>a;
        }
        ijvm_push (i, b );
        break;
```

Og her kommer tilføjelserne til ijvm.spec:

```
0x68 imul
0x6C idiv
0xA1 if_icmplt
0x78 ishl
0x7A ishr
0x7C iushr
```

Tilføjelser til ijvm-util.h:

```
#define IJVM_OPCODE_IMUL          0x68
#define IJVM_OPCODE_IF_ICMPLT     0xA1
#define IJVM_OPCODE_IDIV          0x6C
#define IJVM_OPCODE_ISHL          0x78
#define IJVM_OPCODE_ISHR          0x7A
#define IJVM_OPCODE_IUSHR         0x7C
```

B ijvm2.mal

Mikroprogrammet `ijvm.mal` er blevet ændret ved at følgende er blevet tilføjet:

Her er de tilføjelser vi har gjort til ijvm.mal:

```
# Made by US

# MDR = z, TOS = x, OPC = y
imul = 0x68:
        MAR = SP = SP - 1; rd
        H = TOS
        OPC = MDR
        MDR = 0
imul_test:
        Z = TOS; if (Z) goto imul_end; else goto imul_while
imul_while:
        H = 1
        Z = H AND TOS; if (Z) goto imul_lige; else goto imul_ulige
imul_lige:
        N = TOS = TOS >> 1; if(N) goto imul_fix; else goto imul_cont
imul_fix:
        H = 1 << 8
        H = H << 8
        H = H >> 1
        H = H << 8
        H = H << 8
        TOS = TOS + H
imul_cont:
        H = OPC
        OPC = OPC + H; goto imul_test
imul_ulige:
        H = OPC
        MDR = MDR + H; goto imul_lige
imul_end:
        TOS = MDR; wr; goto main
```

```
idiv =  0x6c:
        MAR = SP = SP - 1; rd
        H = LV
        OPC = MDR
        MDR = H; wr
        MAR = SP = SP + 1
        H = LV = TOS
        MDR = CPP; wr
        empty
        MDR = CPP = 0
## test for nævner er nul
idiv_test_nul:
        Z = TOS; if (Z) goto idiv_nul; else goto idiv_test_equal
idiv_test_equal:
        Z = OPC - H; if (Z) goto idiv_facit1; else goto idiv_test_neg
idiv_test_neg:
        N = TOS; if (N) goto idiv_neg; else goto idiv_test1

idiv_neg:
        N = TOS = LV = H = -H; if (N) goto idiv_nul_branch; else goto idiv_neg_cont
idiv_nul_branch:
        goto idiv_nul
idiv_neg_cont:
        CPP = 1
        MDR = 0; goto idiv_test1

idiv_test1:
        N = H = H + TOS; if (N) goto idiv_while1_end; else goto idiv_while1
idiv_while1:
        TOS = H; goto idiv_test1
idiv_while1_end:
        H = TOS; goto idiv_test2

idiv_test2:
        N = OPC; if (N) goto idiv_neg2; else goto idiv_while3
idiv_neg2:
        H = OPC
        N = OPC = -H; if (N) goto idiv_neg3; else goto idiv_neg4
idiv_neg4:
        CPP = CPP - 1; goto idiv_while3
idiv_neg3:
        ##Tager højde for -2^31/w
        CPP = CPP - 1
```

```
        H = TOS
        MDR = MDR + 1
        OPC = OPC - H; goto idiv_while3
idiv_test4:
        H = TOS
        Z = LV - H; if (Z) goto idiv_end2; else goto idiv_st3
idiv_st3:
        TOS = TOS >> 1
        H = MDR
        MDR = MDR + H
idiv_while3:
        H = TOS
        N = OPC - H; if (N) goto idiv_st2; else goto idiv_st
idiv_st2:
        H = TOS; goto idiv_test4
idiv_st:
        MDR = MDR + 1
        OPC = OPC - H; goto idiv_test4
idiv_facit1:
        MDR = 1; goto idiv_end2
idiv_nul:
        goto idiv_end2
idiv_end2:
        H = MDR; rd
        Z = CPP; if (Z) goto idiv_finish_end; else goto idiv_invert
idiv_invert:
        H = - H; goto idiv_finish_end
idiv_finish_end:
        MAR = SP = SP - 1
        CPP = MDR; rd
        empty
        LV = MDR
        TOS = MDR = H; wr; goto main
```

```
ishl = 0x78:
        MAR = SP = SP - 1; rd
        OPC = 1 << 8
        OPC = OPC >> 1
        OPC = OPC >> 1
        OPC = OPC >> 1
        H = OPC = OPC - 1
        TOS = TOS AND H
ishl_test:
        TOS = TOS - 1; if (N) goto ishl_end; else goto ishl_st
ishl_st:
        H = MDR
        MDR = MDR + H; goto ishl_test
ishl_end:
        TOS = MDR; wr
        goto main

####################################################

ishr = 0x7A:
        MAR = SP = SP - 1; rd
        OPC = 1 << 8
        OPC = OPC >> 1
        OPC = OPC >> 1
        OPC = OPC >> 1
        H = OPC - 1
        TOS = H AND TOS
ishr_while:
        TOS = TOS - 1; if (N) goto ishr_endwhile; else goto ishr_do
ishr_do:
        MDR = MDR >> 1; goto ishr_while
ishr_endwhile:
        TOS = MDR; wr; goto main
```

```
iushr = 0x7c:
        MAR = SP = SP - 1; rd
        OPC = 1 << 8
        OPC = OPC >> 1
        OPC = OPC >> 1
        OPC = OPC >> 1
        H = OPC - 1
        TOS = H AND TOS
iushr_test:
        Z = TOS; if (Z) goto iushr_end; else goto iushr_st
iushr_st:
        MDR = MDR >> 1
        TOS = TOS - 1
        N = MDR; if (N) goto iushr_st2; else goto iushr_test
iushr_st2:
        H = 1 << 8
        H = H << 8
        H = H << 8
        H = H >> 1
        H = H << 8
        MDR = MDR - H; goto iushr_test
iushr_end:
        TOS = MDR; wr; goto main

#############################################
if_icmplt = 0xA1:
        MAR = SP = SP - 1; rd
        MAR = SP = SP - 1
        OPC = MDR; rd; if (N) goto xNeg; else goto xPos
xPos:
        H = TOS; if (N) goto if_icmpltF; else goto if_icmpltPOS
xNeg:
        H = TOS; if (N) goto if_icmpltNEG; else goto if_icmpltT
if_icmpltT:
        TOS = MDR; goto T
if_icmpltF:
        TOS = MDR; goto F
if_icmpltPOS:
        TOS = MDR; goto if_icmplt_end
if_icmpltNEG:
        TOS = MDR; goto if_icmplt_end
if_icmplt_end:
        N = OPC - H; if (N) goto T; else goto F
```

C Afprøvning af gamle ordrer

Koden for test.j:

```
.method main                // int main
.args 3                     // ( int a, int b )
.define a = 1
.define b = 2
                            // {
        bipush 88
// Push object reference.
        iload a
        iload b
        invokevirtual min
        ireturn             // return min ( a, b );
                            // }

.method min                 // int min
.args 3                     // ( int a, int b ){
.define a = 1
.define b = 2
.locals 1                   // int r;
.define r = 3

        iload a             // if ( a >= b )
        iload b
        isub

// stack = a - b, ... ; a - b < 0 => a < b

        iflt else

        iload b             //   r = b;
        istore r
        goto end_if

else:
        iload a             //   r = a;
        istore r

end_if:
        iload r             // return r;
        ireturn
```

```
                              // }
```

Her er så vores mic1 trace:

```
Mic1 Trace of ../ijvm.mic1 with test2.bc Tue Dec 18 01:12:21 2001

                                stack = 0, 1, 82, 17, 15
bipush 88           [10 58]     stack = 88, 0, 1, 82, 17, 15
iload 1             [15 01]     stack = 17, 88, 0, 1, 82, 17, 15
iload 2             [15 02]     stack = 82, 17, 88, 0, 1, 82, 17, 15
invokevirtual 1     [b6 00 01]  stack = 12, 13, 0, 82, 17, 21, 0, 1
iload 1             [15 01]     stack = 17, 12, 13, 0, 82, 17, 21, 0
iload 2             [15 02]     stack = 82, 17, 12, 13, 0, 82, 17, 21
isub                [64]        stack = -65, 12, 13, 0, 82, 17, 21, 0
iflt 10             [9b 00 0a]  stack = 12, 13, 0, 82, 17, 21, 0, 1
iload 1             [15 01]     stack = 17, 12, 13, 0, 82, 17, 21, 0
istore 3            [36 03]     stack = 12, 13, 17, 82, 17, 21, 0, 1
iload 3             [15 03]     stack = 17, 12, 13, 17, 82, 17, 21, 0
ireturn             [ac]        stack = 17, 0, 1, 82, 17, 15
ireturn             [ac]        stack = 17
return value: 17
```

Ved kørsel af diff på trace fra hhv. mic1 og ijvm fik vi:

```
1c1
< IJVM Trace of /users/smunk/ijvm/tests/test2.bc Tue Dec 18 01:12:14 2001
---
> Mic1 Trace of ../ijvm.mic1 with test2.bc Tue Dec 18 01:12:21 2001
```

Forskellene er ganske enkelt overskriften som ved den ene hedder ijvm trace og den anden mic1 trace, og så er stien forskellig.

D Afprøvning af nye ordrer

Først en test af idiv, her testprogrammet:

```
.method main
.args   1
.define OBJREF = 42
.define MAXINT = 2147483647
.define MININT = -2147483648
                                        stack = 0, 1, 55
        bipush 42           [10 2a]     stack = 42, 0, 1, 55
        bipush 42           [10 2a]     stack = 42, 42, 0, 1, 55
        bipush 1            [10 01]     stack = 1, 42, 42, 0, 1, 55
        idiv                [6c]        stack = 42, 42, 0, 1, 55
        pop                 [57]        stack = 42, 0, 1, 55
        bipush 42           [10 2a]     stack = 42, 42, 0, 1, 55
        bipush -9           [10 f7]     stack = -9, 42, 42, 0, 1, 55
        idiv                [6c]        stack = -4, 42, 0, 1, 55
        pop                 [57]        stack = 42, 0, 1, 55
        bipush 42           [10 2a]     stack = 42, 42, 0, 1, 55
        bipush 1            [10 01]     stack = 1, 42, 42, 0, 1, 55
        idiv                [6c]        stack = 42, 42, 0, 1, 55
        pop                 [57]        stack = 42, 0, 1, 55
        ldc_w MININT        [13 00 01]  stack = -2147483648, 42, 0, 1, 55
        ldc_w MININT        [13 00 01]  stack = -2147483648, -2147483648, 42, 0, 1, 55
        idiv                [6c]        stack = 1, 42, 0, 1, 55
        pop                 [57]        stack = 42, 0, 1, 55
        bipush 42           [10 2a]     stack = 42, 42, 0, 1, 55
        bipush 2            [10 02]     stack = 2, 42, 42, 0, 1, 55
        idiv                [6c]        stack = 21, 42, 0, 1, 55
        pop                 [57]        stack = 42, 0, 1, 55
        bipush 42           [10 2a]     stack = 42, 42, 0, 1, 55
        bipush -2           [10 fe]     stack = -2, 42, 42, 0, 1, 55
        idiv                [6c]        stack = -21, 42, 0, 1, 55
        pop                 [57]        stack = 42, 0, 1, 55
        bipush 42           [10 2a]     stack = 42, 42, 0, 1, 55
        ldc_w MININT        [13 00 01]  stack = -2147483648, 42, 42, 0, 1, 55
        idiv                [6c]        stack = 0, 42, 0, 1, 55
        pop                 [57]        stack = 42, 0, 1, 55
        bipush 42           [10 2a]     stack = 42, 42, 0, 1, 55
        bipush 2            [10 02]     stack = 2, 42, 42, 0, 1, 55
        idiv                [6c]        stack = 21, 42, 0, 1, 55
        pop                 [57]        stack = 42, 0, 1, 55
        bipush 42           [10 2a]     stack = 42, 42, 0, 1, 55
```

```
ldc_w MAXINT        [13 00 02]  stack = 2147483647, 42, 42, 0, 1, 55
idiv                [6c]        stack = 0, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
bipush 42           [10 2a]     stack = 42, 42, 0, 1, 55
bipush 4            [10 04]     stack = 4, 42, 42, 0, 1, 55
idiv                [6c]        stack = 10, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
bipush -42          [10 d6]     stack = -42, 42, 0, 1, 55
bipush 7            [10 07]     stack = 7, -42, 42, 0, 1, 55
idiv                [6c]        stack = -6, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
ldc_w MININT        [13 00 01]  stack = -2147483648, 42, 0, 1, 55
ldc_w MAXINT        [13 00 02]  stack = 2147483647, -2147483648, 42, 0, 1, 55
idiv                [6c]        stack = -1, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
ldc_w MAXINT        [13 00 02]  stack = 2147483647, 42, 0, 1, 55
bipush 1            [10 01]     stack = 1, 2147483647, 42, 0, 1, 55
idiv                [6c]        stack = 2147483647, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
bipush -42          [10 d6]     stack = -42, 42, 0, 1, 55
bipush 2            [10 02]     stack = 2, -42, 42, 0, 1, 55
idiv                [6c]        stack = -21, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
bipush 1            [10 01]     stack = 1, 42, 0, 1, 55
bipush 2            [10 02]     stack = 2, 1, 42, 0, 1, 55
idiv                [6c]        stack = 0, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
bipush 3            [10 03]     stack = 3, 42, 0, 1, 55
bipush -9           [10 f7]     stack = -9, 3, 42, 0, 1, 55
idiv                [6c]        stack = 0, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
bipush -1           [10 ff]     stack = -1, 42, 0, 1, 55
bipush 1            [10 01]     stack = 1, -1, 42, 0, 1, 55
idiv                [6c]        stack = -1, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
ldc_w MAXINT        [13 00 02]  stack = 2147483647, 42, 0, 1, 55
ldc_w MAXINT        [13 00 02]  stack = 2147483647, 2147483647, 42, 0, 1, 55
idiv                [6c]        stack = 1, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
ldc_w MININT        [13 00 01]  stack = -2147483648, 42, 0, 1, 55
ldc_w MAXINT        [13 00 02]  stack = 2147483647, -2147483648, 42, 0, 1, 55
idiv                [6c]        stack = -1, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
```

```
bipush 42            [10 2a]     stack = 42, 42, 0, 1, 55
ldc_w MININT         [13 00 01]  stack = -2147483648, 42, 42, 0, 1, 55
idiv                 [6c]        stack = 0, 42, 0, 1, 55
pop                  [57]        stack = 42, 0, 1, 55
bipush -42           [10 d6]     stack = -42, 42, 0, 1, 55
ldc_w MININT         [13 00 01]  stack = -2147483648, -42, 42, 0, 1, 55
idiv                 [6c]        stack = 0, 42, 0, 1, 55
pop                  [57]        stack = 42, 0, 1, 55
ldc_w MAXINT         [13 00 02]  stack = 2147483647, 42, 0, 1, 55
ldc_w MININT         [13 00 01]  stack = -2147483648, 2147483647, 42, 0, 1, 55
idiv                 [6c]        stack = 0, 42, 0, 1, 55
pop                  [57]        stack = 42, 0, 1, 55
ldc_w MININT         [13 00 01]  stack = -2147483648, 42, 0, 1, 55
ldc_w MININT         [13 00 01]  stack = -2147483648, -2147483648, 42, 0, 1, 55
idiv                 [6c]        stack = 1, 42, 0, 1, 55
pop                  [57]        stack = 42, 0, 1, 55
bipush 42            [10 2a]     stack = 42, 42, 0, 1, 55
ldc_w 3              [13 00 03]  stack = 512, 42, 42, 0, 1, 55
idiv                 [6c]        stack = 0, 42, 0, 1, 55
pop                  [57]        stack = 42, 0, 1, 55
bipush -42           [10 d6]     stack = -42, 42, 0, 1, 55
ldc_w 3              [13 00 03]  stack = 512, -42, 42, 0, 1, 55
idiv                 [6c]        stack = 0, 42, 0, 1, 55
pop                  [57]        stack = 42, 0, 1, 55
ldc_w MAXINT         [13 00 02]  stack = 2147483647, 42, 0, 1, 55
ldc_w 3              [13 00 03]  stack = 512, 2147483647, 42, 0, 1, 55
idiv                 [6c]        stack = 4194303, 42, 0, 1, 55
pop                  [57]        stack = 42, 0, 1, 55
ldc_w MININT         [13 00 01]  stack = -2147483648, 42, 0, 1, 55
ldc_w 3              [13 00 03]  stack = 512, -2147483648, 42, 0, 1, 55
idiv                 [6c]        stack = -4194304, 42, 0, 1, 55
pop                  [57]        stack = 42, 0, 1, 55
ldc_w MININT         [13 00 01]  stack = -2147483648, 42, 0, 1, 55
bipush -16           [10 f0]     stack = -16, -2147483648, 42, 0, 1, 55
idiv                 [6c]        stack = 134217728, 42, 0, 1, 55
bipush 0             [10 00]     stack = 0, 134217728, 42, 0, 1, 55
ireturn              [ac]        stack = 0
return value: 0
```

Trace fra mic1:

Mic1 Trace of ../ijvm.mic1 with idiv.bc Tue Dec 18 00:12:51 2001

```
                                stack = 0, 1, 55
bipush 42           [10 2a]     stack = 42, 0, 1, 55
bipush 42           [10 2a]     stack = 42, 42, 0, 1, 55
bipush 1            [10 01]     stack = 1, 42, 42, 0, 1, 55
idiv                [6c]        stack = 42, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
bipush 42           [10 2a]     stack = 42, 42, 0, 1, 55
bipush -9           [10 f7]     stack = -9, 42, 42, 0, 1, 55
idiv                [6c]        stack = -4, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
bipush 42           [10 2a]     stack = 42, 42, 0, 1, 55
bipush 1            [10 01]     stack = 1, 42, 42, 0, 1, 55
idiv                [6c]        stack = 42, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
ldc_w 1             [13 00 01]  stack = -2147483648, 42, 0, 1, 55
ldc_w 1             [13 00 01]  stack = -2147483648, -2147483648, 42, 0, 1, 55
idiv                [6c]        stack = 1, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
bipush 42           [10 2a]     stack = 42, 42, 0, 1, 55
bipush 2            [10 02]     stack = 2, 42, 42, 0, 1, 55
idiv                [6c]        stack = 21, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
bipush 42           [10 2a]     stack = 42, 42, 0, 1, 55
bipush -2           [10 fe]     stack = -2, 42, 42, 0, 1, 55
idiv                [6c]        stack = -21, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
bipush 42           [10 2a]     stack = 42, 42, 0, 1, 55
ldc_w 1             [13 00 01]  stack = -2147483648, 42, 42, 0, 1, 55
idiv                [6c]        stack = 0, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
bipush 42           [10 2a]     stack = 42, 42, 0, 1, 55
bipush 2            [10 02]     stack = 2, 42, 42, 0, 1, 55
idiv                [6c]        stack = 21, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
bipush 42           [10 2a]     stack = 42, 42, 0, 1, 55
ldc_w 2             [13 00 02]  stack = 2147483647, 42, 42, 0, 1, 55
idiv                [6c]        stack = 0, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
bipush 42           [10 2a]     stack = 42, 42, 0, 1, 55
```

```
bipush 4            [10 04]      stack = 4, 42, 42, 0, 1, 55
idiv                [6c]         stack = 10, 42, 0, 1, 55
pop                 [57]         stack = 42, 0, 1, 55
bipush -42          [10 d6]      stack = -42, 42, 0, 1, 55
bipush 7            [10 07]      stack = 7, -42, 42, 0, 1, 55
idiv                [6c]         stack = -6, 42, 0, 1, 55
pop                 [57]         stack = 42, 0, 1, 55
ldc_w 1             [13 00 01]   stack = -2147483648, 42, 0, 1, 55
ldc_w 2             [13 00 02]   stack = 2147483647, -2147483648, 42, 0, 1, 55
idiv                [6c]         stack = -1, 42, 0, 1, 55
pop                 [57]         stack = 42, 0, 1, 55
ldc_w 2             [13 00 02]   stack = 2147483647, 42, 0, 1, 55
bipush 1            [10 01]      stack = 1, 2147483647, 42, 0, 1, 55
idiv                [6c]         stack = 2147483647, 42, 0, 1, 55
pop                 [57]         stack = 42, 0, 1, 55
bipush -42          [10 d6]      stack = -42, 42, 0, 1, 55
bipush 2            [10 02]      stack = 2, -42, 42, 0, 1, 55
idiv                [6c]         stack = -21, 42, 0, 1, 55
pop                 [57]         stack = 42, 0, 1, 55
bipush 1            [10 01]      stack = 1, 42, 0, 1, 55
bipush 2            [10 02]      stack = 2, 1, 42, 0, 1, 55
idiv                [6c]         stack = 0, 42, 0, 1, 55
pop                 [57]         stack = 42, 0, 1, 55
bipush 3            [10 03]      stack = 3, 42, 0, 1, 55
bipush -9           [10 f7]      stack = -9, 3, 42, 0, 1, 55
idiv                [6c]         stack = 0, 42, 0, 1, 55
pop                 [57]         stack = 42, 0, 1, 55
bipush -1           [10 ff]      stack = -1, 42, 0, 1, 55
bipush 1            [10 01]      stack = 1, -1, 42, 0, 1, 55
idiv                [6c]         stack = -1, 42, 0, 1, 55
pop                 [57]         stack = 42, 0, 1, 55
ldc_w 2             [13 00 02]   stack = 2147483647, 42, 0, 1, 55
ldc_w 2             [13 00 02]   stack = 2147483647, 2147483647, 42, 0, 1, 55
idiv                [6c]         stack = 1, 42, 0, 1, 55
pop                 [57]         stack = 42, 0, 1, 55
ldc_w 1             [13 00 01]   stack = -2147483648, 42, 0, 1, 55
ldc_w 2             [13 00 02]   stack = 2147483647, -2147483648, 42, 0, 1, 55
idiv                [6c]         stack = -1, 42, 0, 1, 55
pop                 [57]         stack = 42, 0, 1, 55
bipush 42           [10 2a]      stack = 42, 42, 0, 1, 55
ldc_w 1             [13 00 01]   stack = -2147483648, 42, 42, 0, 1, 55
idiv                [6c]         stack = 0, 42, 0, 1, 55
pop                 [57]         stack = 42, 0, 1, 55
```

```
bipush -42          [10 d6]     stack = -42, 42, 0, 1, 55
ldc_w 1             [13 00 01]  stack = -2147483648, -42, 42, 0, 1, 55
idiv                [6c]        stack = 0, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
ldc_w 2             [13 00 02]  stack = 2147483647, 42, 0, 1, 55
ldc_w 1             [13 00 01]  stack = -2147483648, 2147483647, 42, 0, 1, 55
idiv                [6c]        stack = 0, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
ldc_w 1             [13 00 01]  stack = -2147483648, 42, 0, 1, 55
ldc_w 1             [13 00 01]  stack = -2147483648, -2147483648, 42, 0, 1, 55
idiv                [6c]        stack = 1, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
bipush 42           [10 2a]     stack = 42, 42, 0, 1, 55
ldc_w 3             [13 00 03]  stack = 512, 42, 42, 0, 1, 55
idiv                [6c]        stack = 0, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
bipush -42          [10 d6]     stack = -42, 42, 0, 1, 55
ldc_w 3             [13 00 03]  stack = 512, -42, 42, 0, 1, 55
idiv                [6c]        stack = 0, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
ldc_w 2             [13 00 02]  stack = 2147483647, 42, 0, 1, 55
ldc_w 3             [13 00 03]  stack = 512, 2147483647, 42, 0, 1, 55
idiv                [6c]        stack = 4194303, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
ldc_w 1             [13 00 01]  stack = -2147483648, 42, 0, 1, 55
ldc_w 3             [13 00 03]  stack = 512, -2147483648, 42, 0, 1, 55
idiv                [6c]        stack = -4194304, 42, 0, 1, 55
pop                 [57]        stack = 42, 0, 1, 55
ldc_w 1             [13 00 01]  stack = -2147483648, 42, 0, 1, 55
bipush -16          [10 f0]     stack = -16, -2147483648, 42, 0, 1, 55
idiv                [6c]        stack = 134217728, 42, 0, 1, 55
bipush 0            [10 00]     stack = 0, 134217728, 42, 0, 1, 55
ireturn             [ac]        stack = 0
return value: 0
```

Resultat af diff:

```
1c1
< IJVM Trace of /users/smunk/ijvm/tests/idiv.bc Tue Dec 18 00:58:02 2001
---
> Mic1 Trace of ../ijvm.mic1 with idiv.bc Tue Dec 18 00:12:51 2001
```

Forskellen her er overskriften.
Det skal så her tilføjes at denne test ikke tester for division med nul, da ijvm ikke kan håndtere dette, men kaster en exception, vi har valgt i vores implementation til mic1 at den blot smider 0 på stakken, dette håndteres af vores allerførste betingede hop, som, hvis nævneren er lig 0, hopper til slutningen af programmet og lægger 0 på stakken.

Test af imul:

Testkoden:

```
.method main
.args   1
.define OBJREF = 42
        bipush OBJREF           // stack = 42
        bipush 53               // stack = 53, 42
        bipush 3                // stack = 3, 53, 42
        imul                    // stack = 159, 42
        pop                     // stack = 42
        bipush 3                // stack = 3
        bipush 53               // stack = 53, 3
        imul                    // stack = 159, 42
        pop                     // stack = 42
        bipush 53               // stack = 53, 42
        bipush 0                // stack = 0, 53, 42
        imul                    // stack = 0, 42
        pop                     // stack = 42
        bipush 53               // stack = 53, 42
        bipush 2                // stack = 2, 53, 42
        imul                    // stack = 106, 42
        pop                     // stack = 42
        bipush 53               // stack = 53, 42
        bipush -3               // stack = -3, 53, 42
        imul                    // stack = -159, 42
        pop                     // stack = 42
        bipush -1               // stack = -1, 42
        bipush -53              // stack = -53, -1, 42
        imul                    // stack = 53, 42
        bipush -53              // stack = -53, 53, 42
        imul                    // stack = -2809, 42
        bipush 3                // stack = 3, -2809, 42
        imul                    // stack = -8427, 42
        bipush 55               // stack = 55, -8427, 42
        imul                    // stack = -463485, 42
        bipush -55              // stack = -55, -463485, 42
        imul                    // stack = 25491675, 42
        bipush 55               // stack = 55, 25491675, 42
        imul                    // stack = 1402042125, 42
        bipush 55               // stack = 55, 1402042125, 42
        imul                    // stack = -197094453, 42
```

```
bipush 0                // stack = 0, -197094453, 42
imul                    // stack = 0, 42
                        // Negativ overløb
bipush -53              // stack = -53, 0, 42
bipush 3                // stack = 3, -53, 0, 42
imul                    // stack = -159, 0, 42
bipush 3                // stack = 3, -159, 0, 42
imul                    // stack = -477, 0, 42
bipush -3               // stack = -3, -477, 0 , 42
imul                    // stack = 1431, 0, 42
bipush -3               // stack = -3, 1431, 0, 42
imul                    // stack = -4293, 0, 42
bipush 3                // stack = 3, -4293, 0, 42
imul                    // stack = -12879, 0, 42
bipush -53              // stack = -53, -12879, 0, 42
imul                    // stack = 682587, 0, 42
bipush -53              // stack = -53, 682587, 0, 42
imul                    // stack = -36177111, 0, 42
bipush 3                // stack = 3, -36177111, 0, 42
imul                    // stack = -108531333, 0, 42
bipush 55               // stack = 55, -108531333, 0, 42
imul                    // stack = -5969223315, 0, 42
bipush 0                // stack = 0, -5969223315, 0, 42
imul                    // stack = 0, 0, 42
bipush 0                // stack = 0, 0, 0, 42
ireturn                 // return 0
```

Trace af mic1 kørsel:

```
Mic1 Trace of ../ijvm.mic1 with imul.bc Mon Dec 17 22:38:21 2001

                                        stack = 0, 1, 27
bipush 42               [10 2a]         stack = 42, 0, 1, 27
bipush 53               [10 35]         stack = 53, 42, 0, 1, 27
bipush 3                [10 03]         stack = 3, 53, 42, 0, 1, 27
imul                    [68]            stack = 159, 42, 0, 1, 27
pop                     [57]            stack = 42, 0, 1, 27
bipush 3                [10 03]         stack = 3, 42, 0, 1, 27
bipush 53               [10 35]         stack = 53, 3, 42, 0, 1, 27
imul                    [68]            stack = 159, 42, 0, 1, 27
pop                     [57]            stack = 42, 0, 1, 27
bipush 53               [10 35]         stack = 53, 42, 0, 1, 27
bipush 0                [10 00]         stack = 0, 53, 42, 0, 1, 27
imul                    [68]            stack = 0, 42, 0, 1, 27
pop                     [57]            stack = 42, 0, 1, 27
bipush 53               [10 35]         stack = 53, 42, 0, 1, 27
bipush 2                [10 02]         stack = 2, 53, 42, 0, 1, 27
imul                    [68]            stack = 106, 42, 0, 1, 27
pop                     [57]            stack = 42, 0, 1, 27
bipush 53               [10 35]         stack = 53, 42, 0, 1, 27
bipush -3               [10 fd]         stack = -3, 53, 42, 0, 1, 27
imul                    [68]            stack = -159, 42, 0, 1, 27
pop                     [57]            stack = 42, 0, 1, 27
bipush -1               [10 ff]         stack = -1, 42, 0, 1, 27
bipush -53              [10 cb]         stack = -53, -1, 42, 0, 1, 27
imul                    [68]            stack = 53, 42, 0, 1, 27
bipush -53              [10 cb]         stack = -53, 53, 42, 0, 1, 27
imul                    [68]            stack = -2809, 42, 0, 1, 27
bipush 3                [10 03]         stack = 3, -2809, 42, 0, 1, 27
imul                    [68]            stack = -8427, 42, 0, 1, 27
bipush 55               [10 37]         stack = 55, -8427, 42, 0, 1, 27
imul                    [68]            stack = -463485, 42, 0, 1, 27
bipush -55              [10 c9]         stack = -55, -463485, 42, 0, 1, 27
imul                    [68]            stack = 25491675, 42, 0, 1, 27
bipush 55               [10 37]         stack = 55, 25491675, 42, 0, 1, 27
imul                    [68]            stack = 1402042125, 42, 0, 1, 27
bipush 55               [10 37]         stack = 55, 1402042125, 42, 0, 1, 27
imul                    [68]            stack = -197094453, 42, 0, 1, 27
bipush 0                [10 00]         stack = 0, -197094453, 42, 0, 1, 27
imul                    [68]            stack = 0, 42, 0, 1, 27
```

```
bipush -53          [10 cb]     stack = -53, 0, 42, 0, 1, 27
bipush 3            [10 03]     stack = 3, -53, 0, 42, 0, 1, 27
imul                [68]        stack = -159, 0, 42, 0, 1, 27
bipush 3            [10 03]     stack = 3, -159, 0, 42, 0, 1, 27
imul                [68]        stack = -477, 0, 42, 0, 1, 27
bipush -3           [10 fd]     stack = -3, -477, 0, 42, 0, 1, 27
imul                [68]        stack = 1431, 0, 42, 0, 1, 27
bipush -3           [10 fd]     stack = -3, 1431, 0, 42, 0, 1, 27
imul                [68]        stack = -4293, 0, 42, 0, 1, 27
bipush 3            [10 03]     stack = 3, -4293, 0, 42, 0, 1, 27
imul                [68]        stack = -12879, 0, 42, 0, 1, 27
bipush -53          [10 cb]     stack = -53, -12879, 0, 42, 0, 1, 27
imul                [68]        stack = 682587, 0, 42, 0, 1, 27
bipush -53          [10 cb]     stack = -53, 682587, 0, 42, 0, 1, 27
imul                [68]        stack = -36177111, 0, 42, 0, 1, 27
bipush 3            [10 03]     stack = 3, -36177111, 0, 42, 0, 1, 27
imul                [68]        stack = -108531333, 0, 42, 0, 1, 27
bipush 55           [10 37]     stack = 55, -108531333, 0, 42, 0, 1, 27
imul                [68]        stack = -1674256019, 0, 42, 0, 1, 27
bipush 0            [10 00]     stack = 0, -1674256019, 0, 42, 0, 1, 27
imul                [68]        stack = 0, 0, 42, 0, 1, 27
bipush 0            [10 00]     stack = 0, 0, 0, 42, 0, 1, 27
ireturn             [ac]        stack = 0
return value: 0
```

Resultat af diff:

```
1c1
< IJVM Trace of /users/smunk/ijvm/tests/imul.bc Mon Dec 17 22:37:58 2001
---
> Mic1 Trace of ../ijvm.mic1 with imul.bc Mon Dec 17 22:38:21 2001
```

Forskellen er overskriften.

```
Test af ishr:
Testkoden til ishr:

.method main
.args   1
.define OBJREF = 42

        bipush OBJREF         // stack = 42
        bipush 1              // stack = 1, 42
        bipush 0              // stack = 0, 1, 42
        ishr                  // stack = 1, 42
        pop                   // stack = 42
        bipush 1              // stack = 1, 42
        bipush 1              // stack = 1, 1, 42
        ishr                  // stack = 0, 42
        pop                   // stack = 42
        bipush 2              // stack = 2, 42
        bipush 4              // stack = 4, 2, 42
        ishr                  // stack = 0, 42
        pop                   // stack = 42
        bipush 31             // stack = 31, 42
        ldc_w  2147483647     // stack = 2147483647, 31, 42
        ishr                  // stack = 0, 42
        pop                   // stack = 42
        bipush 33             // stack = 33, 42
        bipush 32             // stack = 32, 33, 42
        ishr                  // stack = 33, 42
        pop                   // stack = 42
        bipush 4              // stack = 4, 42
        bipush -1             // stack = -1, 2, 42
        ishr                  // stack = 0, 42
        pop                   // stack = 42
        bipush 1              // stack = 1, 42
        bipush 0              // stack = 0, 3, 42
        ishr                  // stack = 1, 42
        pop                   // stack = 42
        bipush 2              // stack = 2, 42
        ldc_w  -2147483648    // stack = -2147483648, 2, 42
        ishr                  // stack = 2, 42
        pop                   // stack = 42
        bipush 24             // stack = 24, 42
        bipush 3              // stack = 3, 24, 42
        ishr                  // stack = 3, 42
```

```
pop                       // stack = 42
bipush -3                 // stack = -3, 42
bipush 20                 // stack = 20, -3, 42
ishr                      // stack = -1, 42
pop                       // stack = 42
bipush -5                 // stack = -5, 42
ldc_w 2147483647          // stack = 2147483647, -5, 42
ishr                      // stack = -1, 42
pop                       // stack = 42
bipush 0                  // stack = 0, 42
ireturn                   // return 0
```

Her er der så et trace af kørsel på mic1:

```
Mic1 Trace of ../ijvm.mic1 with ishr.bc Mon Dec 17 22:52:11 2001

                                      stack = 0, 1, 24
bipush 42           [10 2a]     stack = 42, 0, 1, 24
bipush 1            [10 01]     stack = 1, 42, 0, 1, 24
bipush 0            [10 00]     stack = 0, 1, 42, 0, 1, 24
ishr                [7a]        stack = 1, 42, 0, 1, 24
pop                 [57]        stack = 42, 0, 1, 24
bipush 1            [10 01]     stack = 1, 42, 0, 1, 24
bipush 1            [10 01]     stack = 1, 1, 42, 0, 1, 24
ishr                [7a]        stack = 0, 42, 0, 1, 24
pop                 [57]        stack = 42, 0, 1, 24
bipush 2            [10 02]     stack = 2, 42, 0, 1, 24
bipush 4            [10 04]     stack = 4, 2, 42, 0, 1, 24
ishr                [7a]        stack = 0, 42, 0, 1, 24
pop                 [57]        stack = 42, 0, 1, 24
bipush 31           [10 1f]     stack = 31, 42, 0, 1, 24
ldc_w 1             [13 00 01]  stack = 2147483647, 31, 42, 0, 1, 24
ishr                [7a]        stack = 0, 42, 0, 1, 24
pop                 [57]        stack = 42, 0, 1, 24
bipush 33           [10 21]     stack = 33, 42, 0, 1, 24
bipush 32           [10 20]     stack = 32, 33, 42, 0, 1, 24
ishr                [7a]        stack = 33, 42, 0, 1, 24
pop                 [57]        stack = 42, 0, 1, 24
bipush 4            [10 04]     stack = 4, 42, 0, 1, 24
bipush -1           [10 ff]     stack = -1, 4, 42, 0, 1, 24
ishr                [7a]        stack = 0, 42, 0, 1, 24
pop                 [57]        stack = 42, 0, 1, 24
bipush 1            [10 01]     stack = 1, 42, 0, 1, 24
bipush 0            [10 00]     stack = 0, 1, 42, 0, 1, 24
ishr                [7a]        stack = 1, 42, 0, 1, 24
pop                 [57]        stack = 42, 0, 1, 24
bipush 2            [10 02]     stack = 2, 42, 0, 1, 24
ldc_w 2             [13 00 02]  stack = -2147483648, 2, 42, 0, 1, 24
ishr                [7a]        stack = 2, 42, 0, 1, 24
pop                 [57]        stack = 42, 0, 1, 24
bipush 24           [10 18]     stack = 24, 42, 0, 1, 24
bipush 3            [10 03]     stack = 3, 24, 42, 0, 1, 24
ishr                [7a]        stack = 3, 42, 0, 1, 24
pop                 [57]        stack = 42, 0, 1, 24
bipush -3           [10 fd]     stack = -3, 42, 0, 1, 24
```

```
bipush 20          [10 14]     stack = 20, -3, 42, 0, 1, 24
ishr               [7a]        stack = -1, 42, 0, 1, 24
pop                [57]        stack = 42, 0, 1, 24
bipush -5          [10 fb]     stack = -5, 42, 0, 1, 24
ldc_w 1            [13 00 01]  stack = 2147483647, -5, 42, 0, 1, 24
ishr               [7a]        stack = -1, 42, 0, 1, 24
pop                [57]        stack = 42, 0, 1, 24
bipush 0           [10 00]     stack = 0, 42, 0, 1, 24
ireturn            [ac]        stack = 0
return value: 0
```

Her er resultatet af kørsel af diff:

```
1c1
< IJVM Trace of /users/smunk/ijvm/tests/ishr.bc Mon Dec 17 22:52:29 2001
---
> Mic1 Trace of ../ijvm.mic1 with ishr.bc Mon Dec 17 22:52:11 2001
```

Forskellen er overskriften på de to tracefiler.

Test af ishl:
Testkoden til ishl:

```
.method main
.args   1
.define OBJREF = 42

         bipush OBJREF          // stack = 42
         bipush 1               // stack = 1, 42
         bipush 1               // stack = 1, 1, 42
         ishl                   // stack = 2, 42
         pop                    // stack = 42
         bipush 0               // stack = 0, 42
         bipush 1               // stack = 1, 0, 42
         ishl                   // stack = 0, 42
         pop                    // stack = 42
         bipush 1               // stack = 1, 42
         bipush 0               // stack = 0, 1, 42
         ishl                   // stack = 1, 42
         pop                    // stack = 42
         bipush -1              // stack = -1, 42
         bipush 1               // stack = 1, -1, 42
         ishl                   // stack = -2, 42
         pop                    // stack = 42
         bipush 1               // stack = 1, 42
         bipush -1              // stack = -1, 1, 42
         ishl                   // stack = -2147483648, 42
         pop                    // stack = 42
                                // 2^26 = 67108864
         ldc_w 67108864         // stack = 67108864, 42
         bipush 5               // stack = 5, 67108864, 42
         ishl                   // stack = -2147483648, 42
         pop                    // stack = 42
                                // 2^25 = 33554432
         ldc_w 33554432         // stack = 33554432, 42
         bipush 5               // stack = 5, 33554432, 42
         ishl                   // stack = 1073741824, 42
         pop                    // stack = 42
         bipush 1               // stack = 1, 42
         bipush 31              // stack = 31, 1, 42
         ishl                   // stack = -2147483648, 42
         pop                    // stack = 42
         bipush 1               // stack = 1, 42
```

```
bipush 32              // stack = 32, 1, 42
ishl                   // stack = 1, 42
pop                    // stack = 42
bipush 1               // stack = 1, 42
bipush -31             // stack = -31, 1, 42
ishl                   // stack = 2, 42
pop                    // stack = 42
bipush 1               // stack = 1, 42
bipush -32             // stack = -32, 1, 42
ishl                   // stack = 1, 42
pop                    // stack = 42
bipush 2               // stack = 2, 42
bipush 0               // stack = 0, 2, 42
ishl                   // stack = 2, 42
pop                    // stack = 42
bipush 0               // stack = 0, 42
bipush 0               // stack = 0, 0, 42
ishl                   // stack = 0, 42
pop                    // stack = 42
bipush -2              // stack = -2, 42
bipush 0               // stack = 0, -2, 42
ishl                   // stack = -2, 42
pop                    // stack = 42
bipush 0               // stack = 0, 42
bipush -3              // stack = -3, 0, 42
ishl                   // stack = 0, 42
pop                    // stack = 42
bipush -1              // stack = -1, 42
bipush -1              // stack = -1, -1, 42
ishl                   // stack = -2147483648, 42
pop                    // stack = 42
bipush 1               // stack = 1, 42
bipush 2               // stack = 2, 1, 42
ishl                   // stack = -2147483648, 42
pop                    // stack = 42

bipush 0               // stack = 0
ireturn                // return 0
```

Trace af kørsel på mic1:

```
Mic1 Trace of ../../ijvm.mic1 with ../ishl.bc Tue Dec 18 02:26:31 2001

                                    stack = 0, 1, 33
bipush 42           [10 2a]         stack = 42, 0, 1, 33
bipush 1            [10 01]         stack = 1, 42, 0, 1, 33
bipush 1            [10 01]         stack = 1, 1, 42, 0, 1, 33
ishl                [78]            stack = 2, 42, 0, 1, 33
pop                 [57]            stack = 42, 0, 1, 33
bipush 0            [10 00]         stack = 0, 42, 0, 1, 33
bipush 1            [10 01]         stack = 1, 0, 42, 0, 1, 33
ishl                [78]            stack = 0, 42, 0, 1, 33
pop                 [57]            stack = 42, 0, 1, 33
bipush 1            [10 01]         stack = 1, 42, 0, 1, 33
bipush 0            [10 00]         stack = 0, 1, 42, 0, 1, 33
ishl                [78]            stack = 1, 42, 0, 1, 33
pop                 [57]            stack = 42, 0, 1, 33
bipush -1           [10 ff]         stack = -1, 42, 0, 1, 33
bipush 1            [10 01]         stack = 1, -1, 42, 0, 1, 33
ishl                [78]            stack = -2, 42, 0, 1, 33
pop                 [57]            stack = 42, 0, 1, 33
bipush 1            [10 01]         stack = 1, 42, 0, 1, 33
bipush -1           [10 ff]         stack = -1, 1, 42, 0, 1, 33
ishl                [78]            stack = -2147483648, 42, 0, 1, 33
pop                 [57]            stack = 42, 0, 1, 33
ldc_w 1             [13 00 01]      stack = 67108864, 42, 0, 1, 33
bipush 5            [10 05]         stack = 5, 67108864, 42, 0, 1, 33
ishl                [78]            stack = -2147483648, 42, 0, 1, 33
pop                 [57]            stack = 42, 0, 1, 33
ldc_w 2             [13 00 02]      stack = 33554432, 42, 0, 1, 33
bipush 5            [10 05]         stack = 5, 33554432, 42, 0, 1, 33
ishl                [78]            stack = 1073741824, 42, 0, 1, 33
pop                 [57]            stack = 42, 0, 1, 33
bipush 1            [10 01]         stack = 1, 42, 0, 1, 33
bipush 31           [10 1f]         stack = 31, 1, 42, 0, 1, 33
ishl                [78]            stack = -2147483648, 42, 0, 1, 33
pop                 [57]            stack = 42, 0, 1, 33
bipush 1            [10 01]         stack = 1, 42, 0, 1, 33
bipush 32           [10 20]         stack = 32, 1, 42, 0, 1, 33
ishl                [78]            stack = 1, 42, 0, 1, 33
pop                 [57]            stack = 42, 0, 1, 33
bipush 1            [10 01]         stack = 1, 42, 0, 1, 33
```

```
bipush -31          [10 e1]     stack = -31, 1, 42, 0, 1, 33
ishl                [78]        stack = 2, 42, 0, 1, 33
pop                 [57]        stack = 42, 0, 1, 33
bipush 1            [10 01]     stack = 1, 42, 0, 1, 33
bipush -32          [10 e0]     stack = -32, 1, 42, 0, 1, 33
ishl                [78]        stack = 1, 42, 0, 1, 33
pop                 [57]        stack = 42, 0, 1, 33
bipush 2            [10 02]     stack = 2, 42, 0, 1, 33
bipush 0            [10 00]     stack = 0, 2, 42, 0, 1, 33
ishl                [78]        stack = 2, 42, 0, 1, 33
pop                 [57]        stack = 42, 0, 1, 33
bipush 0            [10 00]     stack = 0, 42, 0, 1, 33
bipush 0            [10 00]     stack = 0, 0, 42, 0, 1, 33
ishl                [78]        stack = 0, 42, 0, 1, 33
pop                 [57]        stack = 42, 0, 1, 33
bipush -2           [10 fe]     stack = -2, 42, 0, 1, 33
bipush 0            [10 00]     stack = 0, -2, 42, 0, 1, 33
ishl                [78]        stack = -2, 42, 0, 1, 33
pop                 [57]        stack = 42, 0, 1, 33
bipush 0            [10 00]     stack = 0, 42, 0, 1, 33
bipush -3           [10 fd]     stack = -3, 0, 42, 0, 1, 33
ishl                [78]        stack = 0, 42, 0, 1, 33
pop                 [57]        stack = 42, 0, 1, 33
bipush -1           [10 ff]     stack = -1, 42, 0, 1, 33
bipush -1           [10 ff]     stack = -1, -1, 42, 0, 1, 33
ishl                [78]        stack = -2147483648, 42, 0, 1, 33
pop                 [57]        stack = 42, 0, 1, 33
bipush 1            [10 01]     stack = 1, 42, 0, 1, 33
bipush 2            [10 02]     stack = 2, 1, 42, 0, 1, 33
ishl                [78]        stack = 4, 42, 0, 1, 33
pop                 [57]        stack = 42, 0, 1, 33
bipush 0            [10 00]     stack = 0, 42, 0, 1, 33
ireturn             [ac]        stack = 0
return value: 0
```

Resultat af kørsel af diff på to trace fra hhv. ijvm og mic1:

```
< Mic1 Trace of ../../ijvm.mic1 with ../ishl.bc Tue Dec 18 02:26:31 2001
---
> IJVM Trace of /users/smunk/ijvm/tests/ishl.bc Tue Dec 18 02:26:28 2001
```

Forskellen er her kun overskriften på tracet.

Test af iushr:
Testkoden til iushr:

```
.method main
.args   1
.define OBJREF = 42

        bipush OBJREF          // stack = 42
        bipush 1               // stack = 1, 42
        bipush 0               // stack = 0, 1, 42
        iushr                  // stack = 0, 42
        pop                    // stack = 42
        bipush 1               // stack = 1, 42
        bipush 1               // stack = 1, 1, 42
        iushr                  // stack = 0, 42
        pop                    // stack = 42
        bipush 2               // stack = 2, 42
        bipush 4               // stack = 4, 2, 42
        iushr                  // stack = 1, 42
        pop                    // stack = 42
        bipush 31              // stack = 31, 42
                               // 2^31-1 = 2147483647
        ldc_w  2147483647      // stack = 2147483647, 31, 42
        iushr                  // stack = 0, 42
        pop                    // stack = 42
        bipush 33              // stack = 33, 42
        bipush 32              // stack = 32, 33, 42
        iushr                  // stack = 16, 42
        pop                    // stack = 42
        bipush 2               // stack = 2, 42
        bipush -4              // stack = -4, 2, 42
                               // 2^30-1 = 1073741823
        iushr                  // stack = 1073741823, 42
        pop                    // stack = 42
        bipush 3               // stack = 3, 42
        bipush -17             // stack = -17, 3, 42
        iushr                  // stack = 536870909, 42
        pop                    // stack = 42
        bipush 12              // stack = 12, 42
                               // -2^31 = -2147483648
        ldc_w  -2147483648     // stack = -2147483648, 12, 42
        iushr                  // stack = 524288, 42
        pop                    // stack = 42
```

```
bipush 24               // stack = 24, 42
bipush -3               // stack = -3, 24, 42
iushr                   // stack = 255, 42
pop                     // stack = 42
bipush -3               // stack = -3, 42
bipush 20               // stack = 20, -3, 42
iushr                   // stack = 0, 42
pop                     // stack = 42
bipush 5                // stack = 5, 42
ldc_w  -257             // stack = -257, 5, 42
iushr                   // stack = 134217719, 42
pop                     // stack = 42
bipush 3                // stack = 3, 42
                        // -2^31 = -2147483648
ldc_w -2147483648       // stack = -2147483648, 3, 42
                        // 2^28 = 268435456
iushr                   // stack = 268435456, 42
pop                     // stack = 42
bipush 0                // stack = 0, 42
ireturn                 // return 0
```

Trace fra kørsel på mic1:

```
Mic1 Trace of ../../ijvm.mic1 with ../iushr.bc Tue Dec 18 02:28:17 2001

                                stack = 0, 1, 27
bipush 42           [10 2a]     stack = 42, 0, 1, 27
bipush 1            [10 01]     stack = 1, 42, 0, 1, 27
bipush 0            [10 00]     stack = 0, 1, 42, 0, 1, 27
iushr               [7c]        stack = 1, 42, 0, 1, 27
pop                 [57]        stack = 42, 0, 1, 27
bipush 1            [10 01]     stack = 1, 42, 0, 1, 27
bipush 1            [10 01]     stack = 1, 1, 42, 0, 1, 27
iushr               [7c]        stack = 0, 42, 0, 1, 27
pop                 [57]        stack = 42, 0, 1, 27
bipush 2            [10 02]     stack = 2, 42, 0, 1, 27
bipush 4            [10 04]     stack = 4, 2, 42, 0, 1, 27
iushr               [7c]        stack = 0, 42, 0, 1, 27
pop                 [57]        stack = 42, 0, 1, 27
bipush 31           [10 1f]     stack = 31, 42, 0, 1, 27
ldc_w 1             [13 00 01]  stack = 2147483647, 31, 42, 0, 1, 27
iushr               [7c]        stack = 0, 42, 0, 1, 27
pop                 [57]        stack = 42, 0, 1, 27
bipush 33           [10 21]     stack = 33, 42, 0, 1, 27
bipush 32           [10 20]     stack = 32, 33, 42, 0, 1, 27
iushr               [7c]        stack = 33, 42, 0, 1, 27
pop                 [57]        stack = 42, 0, 1, 27
bipush 2            [10 02]     stack = 2, 42, 0, 1, 27
bipush -4           [10 fc]     stack = -4, 2, 42, 0, 1, 27
iushr               [7c]        stack = 0, 42, 0, 1, 27
pop                 [57]        stack = 42, 0, 1, 27
bipush 3            [10 03]     stack = 3, 42, 0, 1, 27
bipush -17          [10 ef]     stack = -17, 3, 42, 0, 1, 27
iushr               [7c]        stack = 0, 42, 0, 1, 27
pop                 [57]        stack = 42, 0, 1, 27
bipush 12           [10 0c]     stack = 12, 42, 0, 1, 27
ldc_w 2             [13 00 02]  stack = -2147483648, 12, 42, 0, 1, 27
iushr               [7c]        stack = 12, 42, 0, 1, 27
pop                 [57]        stack = 42, 0, 1, 27
bipush 24           [10 18]     stack = 24, 42, 0, 1, 27
bipush -3           [10 fd]     stack = -3, 24, 42, 0, 1, 27
iushr               [7c]        stack = 0, 42, 0, 1, 27
pop                 [57]        stack = 42, 0, 1, 27
bipush -3           [10 fd]     stack = -3, 42, 0, 1, 27
```

```
bipush 20            [10 14]     stack = 20, -3, 42, 0, 1, 27
iushr                [7c]        stack = 4095, 42, 0, 1, 27
pop                  [57]        stack = 42, 0, 1, 27
bipush 5             [10 05]     stack = 5, 42, 0, 1, 27
ldc_w 3              [13 00 03]  stack = -257, 5, 42, 0, 1, 27
iushr                [7c]        stack = 0, 42, 0, 1, 27
pop                  [57]        stack = 42, 0, 1, 27
bipush 3             [10 03]     stack = 3, 42, 0, 1, 27
ldc_w 2              [13 00 02]  stack = -2147483648, 3, 42, 0, 1, 27
iushr                [7c]        stack = 3, 42, 0, 1, 27
pop                  [57]        stack = 42, 0, 1, 27
bipush 0             [10 00]     stack = 0, 42, 0, 1, 27
ireturn              [ac]        stack = 0
return value: 0

Resultat af diff:

1c1
< IJVM Trace of /users/smunk/ijvm/tests/iushr.bc Mon Dec 17 20:10:08 2001
---
> Mic1 Trace of ../ijvm.mic1 with iushr.bc Mon Dec 17 20:19:13 2001

Forskellen er kun overskriften.
```

Test af if_icmplt:
Testkoden til if_icmplt:

```
.method main

bipush 42                       // stack = 42
bipush 4                        // stack = 4, 42
bipush 2                        // stack = 2, 4, 42
if_icmplt ok1                   // stack = 42
fejl1:
bipush 0                        // stack = 0, 42
ok1:
bipush -4                       // stack = -4, 0, 42
bipush 2                        // stack = 2, -4, 0, 42
if_icmplt ok2                   // stack = 0, 42
fejl2:
bipush 0
ok2:
bipush 2                        // stack = 2, 0, 42
bipush 4                        // stack = 4, 2, 0, 42
if_icmplt ok3                   // stack = 0, 42
fejl3:
bipush 0
ok3:
bipush 4                        // stack = 4, 0, 42
bipush -2                       // stack = -2, 4, 0, 42
if_icmplt ok4                   // stack = 0, 42
fejl4:
bipush 0                        // stack = 0 ,0, 42
ok4:
bipush -8                       // stack = -8, 0, 0, 42
bipush 6                        // stack = 6, -8, 0, 0, 42
if_icmplt ok5                   // stack = 0, 0, 42
fejl5:
bipush 0
ok5:
bipush -3                       // stack = -3, 0, 0, 42
bipush -6                       // stack = -6, -3, 0, 0, 42
if_icmplt ok6                   // stack = 0, 0, 42
fejl6:
bipush 0                        // stack = 0, 0, 0, 42
ok6:
bipush 7                        // stack = 7, 0, 0, 0, 42
```

```
bipush 3                        // stack = 3, 7, 0, 0, 0, 42
if_icmplt ok7                   // stack = 0, 0, 0, 42
fejl7:
bipush 0                        // stack = 0, 0, 0, 0, 42
ok7:
bipush -7                       // stack = -7, 0, 0, 0, 0, 42
bipush -3                       // stack = -3, -7, 0, 0, 0, 0, 42
if_icmplt ok8                   // stack = 0, 0, 0, 0, 42
fejl8:
bipush 0
ok8:
bipush 2                        // stack = 2, 0, 0, 0, 0, 42
bipush 1                        // stack = 1, 2, 0, 0, 0, 0, 42
if_icmplt ok9                   // stack = 0, 0, 0, 0, 42
fejl9:
bipush 0                        // stack = 0, 0, 0, 0, 0, 42
ok9:
bipush 0                        // stack = 0, 0, 0, 0, 0, 0, 42
ireturn                         // stack = 0
```

Trace af mic1 kørsel:

```
Mic1 Trace of ../../ijvm.mic1 with ../if_icmplt.bc Tue Dec 18 02:43:17 2001

                                stack = 0, 1, 25
bipush 42           [10 2a]     stack = 42, 0, 1, 25
bipush 4            [10 04]     stack = 4, 42, 0, 1, 25
bipush 2            [10 02]     stack = 2, 4, 42, 0, 1, 25
if_icmplt 5         [a1 00 05]  stack = 42, 0, 1, 25
bipush 0            [10 00]     stack = 0, 42, 0, 1, 25
bipush -4           [10 fc]     stack = -4, 0, 42, 0, 1, 25
bipush 2            [10 02]     stack = 2, -4, 0, 42, 0, 1, 25
if_icmplt 5         [a1 00 05]  stack = 0, 42, 0, 1, 25
bipush 2            [10 02]     stack = 2, 0, 42, 0, 1, 25
bipush 4            [10 04]     stack = 4, 2, 0, 42, 0, 1, 25
if_icmplt 5         [a1 00 05]  stack = 0, 42, 0, 1, 25
bipush 4            [10 04]     stack = 4, 0, 42, 0, 1, 25
bipush -2           [10 fe]     stack = -2, 4, 0, 42, 0, 1, 25
if_icmplt 5         [a1 00 05]  stack = 0, 42, 0, 1, 25
bipush 0            [10 00]     stack = 0, 0, 42, 0, 1, 25
bipush -8           [10 f8]     stack = -8, 0, 0, 42, 0, 1, 25
bipush 6            [10 06]     stack = 6, -8, 0, 0, 42, 0, 1, 25
if_icmplt 5         [a1 00 05]  stack = 0, 0, 42, 0, 1, 25
bipush -3           [10 fd]     stack = -3, 0, 0, 42, 0, 1, 25
bipush -6           [10 fa]     stack = -6, -3, 0, 0, 42, 0, 1, 25
if_icmplt 5         [a1 00 05]  stack = 0, 0, 42, 0, 1, 25
bipush 0            [10 00]     stack = 0, 0, 0, 42, 0, 1, 25
bipush 7            [10 07]     stack = 7, 0, 0, 0, 42, 0, 1, 25
bipush 3            [10 03]     stack = 3, 7, 0, 0, 0, 42, 0, 1
if_icmplt 5         [a1 00 05]  stack = 0, 0, 0, 42, 0, 1, 25
bipush 0            [10 00]     stack = 0, 0, 0, 0, 42, 0, 1, 25
bipush -7           [10 f9]     stack = -7, 0, 0, 0, 0, 42, 0, 1
bipush -3           [10 fd]     stack = -3, -7, 0, 0, 0, 0, 42, 0
if_icmplt 5         [a1 00 05]  stack = 0, 0, 0, 0, 42, 0, 1, 25
bipush 2            [10 02]     stack = 2, 0, 0, 0, 0, 42, 0, 1
bipush 1            [10 01]     stack = 1, 2, 0, 0, 0, 0, 42, 0
if_icmplt 5         [a1 00 05]  stack = 0, 0, 0, 0, 42, 0, 1, 25
bipush 0            [10 00]     stack = 0, 0, 0, 0, 0, 42, 0, 1
bipush 0            [10 00]     stack = 0, 0, 0, 0, 0, 0, 42, 0
ireturn             [ac]        stack = 0
return value: 0
```

Resultatet af dif:

```
1c1
< Mic1 Trace of ../../ijvm.mic1 with ../if_icmplt.bc Tue Dec 18 02:43:17 2001
---
> IJVM Trace of /users/smunk/ijvm/tests/if_icmplt.bc Tue Dec 18 02:42:52 2001
```

Og endnu en gang er forskellen kun i overskriften.